INSTRUCTIONS :

1. Complete all questions in your designated project group.
2. All members must contribute to writing the codes. (i.e. 1 question = 1 person, and share the workload if there's an additional question relative to the actual number of members in your team (i.e. 5)). Ensure that all members must understand and explain codes from any of the questions.
3. During viva, all students in each team will be randomly asked to describe, answer and edit any of the answers provided. Marks will be given to your ability to present the answers.

Lab Report

Prepare a report to solve the above problems. The report should contain all the sections as below for each question:

Section	Description
1. Problem	Description on the problem
2. Solution	Explanation on how to solve the problems below
3. Sample Input & Output	A few sets of input and output (snapshot)
4. Source code	Java source code

Requirements

1. Group Assignment (Grouping is the same as your project group)
2. Cover page that includes all student matric number and full name.
3. Font: Times New Roman 12, Line Spacing: 1 ½ Spacing
4. Submit to Spectrum according to your OCC. **Deadline** : Before your viva session (W9).

Question 1: Anagram Checker

Problem Statement:

During an English club activity, your teacher challenges you to write a program that can find groups of words that are made from the same set of letters, known as anagrams. For example, listen and silent use the same letters but arranged differently.

To make this task easier, you decide to build a Java program that reads several words from the user (consider uppercase and lowercase letters as the same) , removes any duplicates, and automatically groups together words that are anagrams of each other. Words that don't have any matching anagram group will be listed separately at the end.

Your program should first ask the user how many words they want to enter, then read each word. It will compare all the words, identify the anagram groups, display each group with a group number, and finally show the words that do not belong to any group under "Without anagram group."

You must implement and use the following methods:

1. boolean isAnagram(String a, String b)
2. int countChar(String str)

You may **create and use any additional helper methods if necessary.*
You are **not allowed to use ArrayList, HashSet, or any other advanced Java collection classes.*

Input:
The first line contains an integer n, which represents the number of words. The next n lines each contain a word. The words may include uppercase or lowercase letters, they should be treated as the same and duplicate words should be ignored.

Output:
Display all groups of anagrams in the format:

```
Anagram group 1: word1 word2 ...
Anagram group 2: word3 word4 ...
```

Display all remaining words that have no anagram group under:

```
Without anagram group: word5 word6 ...
```

Sample Output:

Input	6 silent apa leng silent listen apa	7 coconut banana nutco moon haiyaa banana ban
Output	Anagram group 1: silent listen Without anagram group: apa leng	Anagram group 1: coconut nutco Anagram group 2: banana ban Without anagram group: moon haiyaa

Question 2 : Number Rearrangement

Problem Statement:

You are helping a local bakery manage their order numbers. Each order has a numeric code, but the bakery wants to know which combination of the digits would give the largest possible order code and the smallest possible order code for record-keeping and sorting purposes. To assist, you decide to write a Java program that takes an order code from the user, filters out any non-digit characters, rearranges the digits to find the largest possible number, and also rearranges them to find the smallest possible number. The program then displays both results so the bakery can quickly organize their orders.

You must implement and use the following methods:

1. String getLargestNum(String num)
2. String getSmallestNum(String num)

You may **create and use any additional helper methods if necessary.*

Input:
The first line contains a string num and may contain digits and non-digit characters (which should be ignored).

Output:
Print the largest and smallest possible number formed by rearranging the digits.

Sample output:

```
Enter number: 12hi978byebye0
Largest number: 987210
Smallest number: 12789
```

```
Enter number: 0025634
Largest number: 6543200
Smallest number: 23456
```

```
Enter number: 19293900
Largest number: 99932100
Smallest number: 123999
```

Question 3: String Analyzer

Problem Statement:

You are required to write a Java program that analyzes a word entered by the user to find:

1. Longest palindrome substring
2. Longest mirrorable substring

A mirrorable string is defined as a string that looks the same when reflected in a mirror, meaning each character has a valid mirror pair while a palindrome string is a word that reads the same forwards and backwards. The program comes with a list of default mirror pairs (e.g., A and A, H and H, b and d, p and q, etc.). The user can also add up to 10 new mirror pairs before testing their input word. Minimum length for the valid palindrome and mirrorable substring is 2. If the string contains multiple longest palindrome substrings or longest mirrorable substrings of the same length, display the first one that appears in the string.

Your program should:

1. Allow the user to input additional mirror pairs, where each pair consists of exactly two characters (e.g., 5 and S, O and 0, 3 and E, etc.).
2. The user may stop entering pairs by pressing Enter on an empty line.
3. Each valid pair should automatically include its reverse form (e.g., entering b d means both b↔d and d↔b are added).
4. Read a string from the user.
5. Determine and display: a. Longest palindrome substring in the string. b. Longest mirrorable substring in the string.
6. If there are multiple longest palindrome or mirrorable substring in the string, display the first one.
7. Display "-" if no palindrome or mirrorable substring exists.

You must implement the following methods:

1. String getLongestPalindrome(String str)
2. String getLongestMirrorWord(String str)

Default Mirror Pairs (19) :

A	H	I	M	O	T	U	V	W	X	Y	o	u	v	w	b	d	p	q
A	H	I	M	O	T	U	V	W	X	Y	o	u	v	w	d	b	q	p

You may **create and use any additional helper methods if necessary.*
You are **not allowed to use ArrayList, HashSet, or any other advanced Java collection classes.*

Input:

Enter up to 10 lines of mirror pairs, each containing two characters separated by a space, an empty line indicates the end of mirror pair input. Then one additional line containing the word to analyze.

Output:
Print the longest palindrome and mirrorable substring in the given string. Display "-" if none are found.

Sample output:

```
Enter additional pairs:
S$
E3

2 pair(s) entered.
Enter word: SMoM$aeioiea
Longest palindrome substring: aeioiea
Longest mirrorable substring: SMoM$
```

```
Enter additional pairs:

0 pair(s) entered.
Enter word: PopoooMNoo
Longest palindrome substring: Pop
Longest mirrorable substring: ooo
```

Question 4: Text Analyzer

Problem statement:

Your grandmother loves writing letters to her old friends, but she often wonders how long her messages are. She can't easily count the words, sentences, or characters by herself. To help her, you decided to write a Java program that can analyze her letters automatically.

The program lets her type her letter line by line and ends when she presses Enter on an empty line. It will then show her the total number of words, characters (with and without spaces), sentences, and even tell her the most common word and the longest word she used.

With this little helper, your grandmother can now focus on writing from her heart while your program does all the counting for her.

You must implement and use the following methods:

1. int wordCount(String str)
2. int characterCount(String str)
3. int characterCountWithoutSpaces(String str)
4. int sentenceCount(String str)
5. String mostFrequentWord(String str)
6. String longestWord(String str)

You may **create and use any additional helper methods if necessary.*
You are **not allowed to use ArrayList, HashSet, or any other advanced Java collection classes.*

Hint:
You may refer to an online **word count tool** or **text analyzer app** to compare and verify your program's output when testing.

Input:
Multiple lines of text representing the letter. The input ends when the user presses **Enter** on an empty line.

Output:
Display the following statistics:

1. Word Count
2. Character Count
3. Character Count without Space
4. Sentence Count
5. Most Frequent Word
6. Longest Word

Sample output:

Input	**Output**
The smell of freshly baked cake filled the entire kitchen. She decorated the chocolate cake with strawberries and cream.	Word Count: 19 Character Count: 120 Character Count without Space: 102 Sentence Count: 2 Most Frequent Word: The Longest Word: strawberries
The wooden table stood in the center of the room, polished until it shone under the soft light. Books, papers, and a steaming cup of coffee were scattered across its surface. Around the table, chairs waited quietly, each with its own place.	Word Count: 42 Character Count: 240 Character Count without Space: 199 Sentence Count: 3 Most Frequent Word: The Longest Word: scattered

Question 5: Lucky Draw

Problem Statement:

During the school carnival, the "Lucky Draw" booth gives students a chance to win prizes by guessing numbers. Each student chooses a target number and buys a certain number of balls, each with a set value. The system randomly generates numbers for all purchased balls. The program should display the generated numbers in a grid where the number of rows and columns equals the ceiling of the square root of the total number of balls, and if there are extra numbers beyond the square of that value, they appear in an additional last row. If the student's target number exactly matches one of the drawn numbers, the program prints a congratulatory message along with the total prize. If the number almost matches (e.g., all but one digit is incorrect), the program notifies them of a near miss along with the near miss numbers. Otherwise, the program tells them to "Try again next time."

The total prize is calculated as:

Total Prize = (Number Length × Ball Value × 1000) / Number of Balls

You are required to write a Java program that allows the student to enter:

1. Number length (3 or 4 digits)
2. Number of balls they want to buy
3. Ball value
4. Target number

The program should generate random numbers for the balls, display them in a grid format(e.g., if the total number of balls is 50, display in 7 columns & 8 rows, because the nearest square root number for 50 is 49 which is 7^2, the remaining number), then check for matches or near misses, and display the corresponding messages and total prize if applicable.

You must implement the program using methods as shown below:

1. int[] generateNum(int numOfBall, int length)
2. void displayNum(int[] num, int numOfBall)
3. boolean foundTarget(int target, int poolNumber)
4. boolean nearMiss(int target, int poolNumber, int length)

You may **create and use any additional helper methods if necessary.*

Input:

1. An integer representing the number length (3 or 4 digits).
2. An integer representing the number of balls.

3. An integer representing the value of each ball.
4. An integer representing the target number.

Output:
Display all generated random numbers in a grid layout. Then display whether the user has:

- ➔ Won (get the exact match number),
- ➔ Almost won (near miss), or
- ➔ Try again next time (no match and no near miss).

If the user wins, show the total prize. Else if a near miss occurs, list the near miss numbers.

Sample Output:

```
Enter number length (3 or 4): 4
Enter number of ball to choose: 100
Enter ball value: 1
Target number: 1234
4713   6740   1110   6346   7797   6715   5134   2047   8850   8091
4970   9881   885    6789   4776   9690   2364   437    220    1801
2363   6030   7181   9353   1057   67     2873   2922   7881   3039
6473   1628   9084   4578   2976   8380   3508   5737   6525   2093
284    7897   5807   6612   3155   7527   3720   5375   4015   343
762    4690   5325   3711   4094   16     6487   6502   6404   2211
5199   5999   8501   4137   4216   620    8308   9819   6996   6321
739    5738   743    2999   6465   3320   3994   6617   5457   2833
5628   8494   2054   9844   179    3299   3098   7217   3920   2756
6110   9041   6752   5687   4860   6973   543    166    4760   6716
Try again next time
```

```
Enter number length (3 or 4): 3
Enter number of ball to choose: 100
Enter ball value: 1
Target number: 123
82     568    83     737    114    682    124    678    129    336
799    985    332    31     699    74     631    485    999    657
622    185    640    212    208    196    466    777    85     48
802    398    648    59     951    173    47     640    65     205
967    858    802    103    597    14     458    374    551    52
214    881    513    657    560    113    764    357    450    690
945    420    123    976    707    403    228    502    689    688
556    830    243    430    95     600    68     140    966    953
430    169    704    770    732    543    143    849    632    209
517    537    317    673    436    769    86     898    357    323
Congratulations!! You Got The Number 123
Total Prize: RM30.00
```

```
Enter number length (3 or 4): 3
Enter number of ball to choose: 105
Enter ball value: 1
Target number: 458
986    624    763    170    862    655    835    354    244    830
325    438    30     20     950    249    522    325    306    230
693    108    75     144    898    879    181    509    185    519
781    160    549    483    271    874    654    32     630    180
709    351    110    603    710    979    693    528    559    517
240    883    15     310    763    187    546    544    832    805
689    617    409    503    606    453    136    904    605    200
579    78     534    653    461    995    550    237    255    580
645    855    558    759    931    896    806    961    284    32
925    546    667    177    378    126    706    36     587    936
807    154    134    996    922
You almost get it
453    558
```

Question 6: Two Zero Four Eight

Problem Statement:

You are required to write a Java program that simulates a single move in the 2048 game on a 4 × 4 grid. The program should read the current grid and a direction (LEFT, RIGHT, UP, or DOWN) from the user, slide all non-zero tiles as far as possible in that direction, merge tiles of the same value, and add a new tile (2 or 4) in a random empty position if any movement occurs. Finally, the updated grid should be displayed.

You must implement the program using methods as shown below:

1. void moveLeft(int[][] input)
2. void moveRight(int[][] input)
3. void moveUp(int[][] input)
4. void moveDown(int[][] input)

You may **create and use any additional helper methods if necessary.*

Input:
Enter a 4 × 4 matrix grid to represent the current numbers in the 2048 game, followed by the direction of the next move. Enter 0 to represent the tile is empty.

Output:
Display the updated grid after performing the move and merging the tiles according to 2048 rules. If a move or merge occurs successfully, randomly generate a new tile (2 or 4) and display it randomly on an empty position (0).

Sample Output:

Input	4 0 0 8 2 4 2 2 0 0 4 2 0 4 8 2 up	8 0 0 4 4 0 4 2 8 2 2 0 4 4 4 4 right	2 2 2 4 8 0 0 4 0 2 4 0 4 2 8 0 Down
Output	4 8 2 8 2 0 4 4 0 0 8 2 2 0 0 0	0 0 8 4 0 2 8 2 0 0 8 4 0 0 8 8	0 2 0 0 2 0 2 0 8 2 4 0 4 4 8 8